整理番号　　発送番号　００９８２７
発送日　平成２１年　１月２７日　頁:　1/　5

引用非特許文献

審判請求の番号	不服２００６－　１２８５２
（特許出願の番号）	（特願２００２－　４２５８９）
起案日	平成２１年　１月２６日
審判長　特許庁審判官	山崎 達也
請求人	インターナショナル・ビジネス・マシーンズ・コーポレーション　様
復代理人弁理士	正林 真之　様

引用文献３

第50回(平成７年前期)全国大会

講演論文集(4)

ソフトウェア
基礎理論
ウィンドウシステム
オペレーティングシステム
データベース・情報検索
プログラミング技術
ヒューマンインタフェース

平成７年３月15日～17日　　於：青山学院大学

社団法人 情報処理学会
Information Processing Society of Japan

情報処理学会第50回（平成７年前期）全国大会

5H－5

SunOSにおけるマルチスレッド・プロセス移送方式の提案

鈴木信雄　　脇　英世
東京電機大学　工学部

1.はじめに

最近のワークステーション用OSには、MachのCThreads、OS内のPThreads、SunOS5.xのThreadsライブラリ、SunOS4.1.xのSunLWPライブラリなど、スレッド機構が用意されているものが多い。このようなスレッド機構は、一つのプロセス内で複数の論理的な制御の流れを実現できる機構で、その軽量性と高速性のために、ネットワーク・ファイル・システムやウインドウ・システムなどに利用されている。また、非同期イベントを同期的に扱うことが可能となるため、より自然なプログラミングが可能となり、様々なアプリケーション・プログラムにおいても使用されてきている。

また、プロセス移送は、負荷分散によるCPU資源の効率的利用や、処理の耐故障性の保証による耐障害機能の向上などを目的として、様々な手法が提案され利用されている。しかし、これまでの手法では、プロセス内で動作しているスレッドについて考慮されたものは少ない[6]。

本論では、マルチスレッド・プロセスを移送する時に必要な手法について述べ、本手法をSunLWPへ実際に適用した結果について報告する。

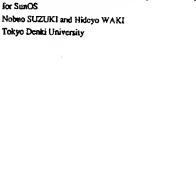

図1　プロセス移送の全体の流れ

Realization of Multi-threaded Process Migration Facility for SunOS
Nobuo SUZUKI and Hideyo WAKI
Tokyo Denki University

2.マルチスレッド・プロセス移送の実現手法

2.1.移送処理全体の流れ

本実現手法の前提となる基本的なプロセス移送の手順は次のとおりである[2]。(図1)

(1)被移送プロセスに対して移送開始をシグナルにより通知する。

(2)シグナルを受け取ったプロセスは、シグナル処理関数中でプロセスのコンテキストを実行可能形式ファイルへ出力し、自分の計算機上のサーバ・プロセス へ移送準備完了の通知を送信する。その後、自分のプロセスを終了する。

(3)サーバ・プロセスは、移送先計算機上のサーバ・プロセスへ移送対象プロセスの実行再開を通知する。

(4)移送先のサーバ・プロセスは、被移送プロセスが出力した実行可能形式ファイルをNFSによりアクセスし、これを起動する。

(5)起動されたプロセスは、(2)で中断されたシグナル処理関数の続きを実行し、その中でプロセスのコンテキストを回復する。

(6)シグナル処理関数から復帰することにより、プロセスが再開される。

この手順は、文献[2]の手法とほぼ同様なものであり、ユーザレベルでの実現が可能である。本論で述べるマルチスレッド対応の手法は、上記手順中のシグナル処理関数内における処理である。

2.2.マルチスレッド・プロセス移送の問題点と対処

移送処理を行うシグナル処理関数は任意の一つのスレッドにおいて実行される。そのため、移送処理中にこのスレッド以外のスレッドが動作してしまうと、一時点のプロセス・コンテキストが取得できない。これに対処するためには、シグナル処理関数の開始時点に、スレッド機構で用意されているsuspend/resume機能を使用して自分以外のスレッドを一時的に停止する必要がある(図2)。その後のプロセス再開時に、シグナル処理関数より復帰する前にsuspendしていた全スレッドのresumeにより、移送前の状態へ戻る。

次に、各スレッドは、一般に固有のスタック領域

予稿集1997-00042-001

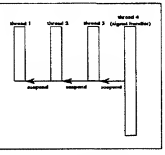

図2　スレッドのsuspend

を有している。このスタック領域は、MachのCThreadsなどはタスクのスタック領域に確保されるが、SunLWPにおいてはプロセスのデータ領域に確保される。どちらの場合においても、スタック領域およびデータ領域をプロセスコンテキスト・ファイルへ保存することにより、スレッド固有のスタック領域を移送する必要がある。

3.マルチスレッド・プロセス移送の実現例

3.1.実現環境

使用した計算機はEthernet LAN上に配置された2台のSunワークステーションを用い、OSはSunOS4.1.2、スレッド機構としてはSunLWPライブラリを使用した。このSunLWPライブラリは、完全な処例性を実現してはおらず、コルーチン形式のスレッド機構を提供している。

3.2.シグナル処理関数内の処理

(1)シグナル処理関数実行スレッドのプライオリティを最高のものにする。

シグナル処理関数実行中に他のスレッドが実行されることによる処理の中断を防ぐために、スレッドのプライオリティを最高のものにする。これには、pod_getmaxpri()により最高プライオリティを取得し、lwp_setpri()でそれをセットすることにより行う。

(2)動作中のスレッドの一時停止

動作中のスレッド識別子はlwp_enumerate()を用いて収集し、lwp_suspend()により一時的に停止状態にする。ただし、この時、lwp_enumerate()で返された全てのスレッドをsuspendすると、自スレッドもsuspendされてしまうため、収集した識別子の最後の識別子(自分自身)は一時停止の対象とはしないようにする。

(3)再起動時には、スタック領域やシグナルマスクの回復処理の後、lwp_resume()により一時停止していたスレッドを再開させる。

(4)最高のプライオリティから以前のプライオリティへ戻す。

4.おわりに

(1)本報告では、マルチスレッド・プロセスを移送する時に必要な注意点を示し、その解決法について述べた。また、本方式をSunOS4.1.2のSunLWPライブラリ上に実装し、正常な処理を確認した。

(2)SunLWPライブラリのエージェントの使用

SunLWPライブラリには、エージェントと呼ばれる非同期イベントをメッセージとして扱える機構が備わっている。しかし、MachやSunOS5.xなどのスレッド機構においてはシグナルをこのようなある特定のスレッドからのメッセージとして処理できる機構は用意されていない。そのために、今回の実装ではより一般的な手法となるように、エージェント機構は使用しなかった。

(3)スレッド用スタックの確保

SunLWPライブラリのlwp_newstk()によって得られるスレッド用スタックは、SunLWPライブラリ内においてしか意味を持たず、ユーザ・プログラムからこの領域を読み出すことはできない。そのため、スレッド用スタックはユーザ・プログラムのデータ領域に確保しておく必要がある。

(4)今後は、SunOS5.xを対象に完全に並列なスレッド機構におけるマルチスレッド・プロセス移送の実装について検討し、さらにスレッド単位での移送手法についても検討する予定である。

参考文献

[1]Sun Microsystems:Programming Utilities and Libraries,Lightweight Processes,March 1990

[2]森山,多田:利用者レベルで実現したプロセス移送ライブラリ,情処OS研究会,1991.7.18

[3]W.R.Stevens:UNIX Network Programming,Prentice Hall,1991

[4]Boykin,Kirschen,Langerman,LoVerso:Programming under Mach,Addison Wesley,1993

[5]SunSoft:Guide to Multithread Programming,Nov. 1993

[6]Milojicic,Giese,Zint:Experiences with Load Distribution on top of the Mach Microkernel,Proc. of the USENIX Symposium, 1993.9

予稿集1997-00042-001

平成七年三月一五日 発行
印刷所 三協印刷株式会社 東京都目黒区本町五ー七
発行所 社団法人 情報処理学会 東京都港区芝浦三ー一六ー二〇（芝浦前川ビル７Ｆ）